

(11)Publication number : 10-214208
(43)Date of publication of application : 11.08.1998

G06F 11/30

(71)Applicant : MEIDENSHA CORP

(72)Inventor : YOSHIDA MIKIO
SUGAYA KATSUHIRO

(57)Abstract:

Figure 1 is a flowchart illustrating the process of determining the number of times a program is executed. The process starts with a block labeled '全体アプリケーション' (Overall Application). This block branches into two paths: '1. アプリケーション' (1. Application) and '2. アプリケーション' (2. Application). Both paths lead to a list of program execution events, which are organized into two columns. The first column contains: 'プロセス情報データベース' (Process Information Database), '起動中フラグ' (Running Flag), '処理中フラグ' (Processing Flag), 'チェックインフラグ' (Check-in Flag), '異常終了フラグ' (Abnormal Termination Flag), '異常フラグ' (Abnormal Flag), 'チェックイン監視期間' (Check-in Monitoring Period), '再起動回数' (Restart Count), 'チェックイン監視カウンタ' (Check-in Monitoring Counter), and '再起動リトライカウンタ' (Restart Retry Counter). The second column contains: '起動中フラグ' (Running Flag), '処理中フラグ' (Processing Flag), 'チェックインフラグ' (Check-in Flag), and '再起動リトライカウンタ' (Restart Retry Counter). The flowchart shows that the process involves checking these flags and counters to determine the execution status and restart count of the application.

SOLUTION: A data base 3 uses an activation flag, processing flag, and check-in flag respectively indicating the activation, processing, and monitoring for each application to be monitored, check-in monitoring cycle for judging the abnormality of the application to be monitored, and check-in processing counter for integrating the number of times of monitoring processing or the like as monitor information. Applications 11-1N to be monitored set and reset each flag according to activation or processing, and a monitoring application 2 detects the abnormality of the application to be monitored by refer to the check-in flag and the count-up or clear of the check-in counter or the like. Also, automatic restoration against temporary abnormality can be attained by the reactivating mechanism of the application after the detection of abnormality.

18.02.2003

[Date of requesting appeal against examiner's

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-214208

(43) 公開日 平成10年(1998) 8月11日

(51) Int.Cl.⁶
G 0 6 F 11/30

識別記号
3 1 0

F I
G 0 6 F 11/30

3 1 0 H

審査請求 未請求 請求項の数2 OL (全 7 頁)

(21) 出願番号 特願平9-17277

(22) 出願日 平成9年(1997) 1月31日

(71) 出願人 000006105

株式会社明電舎

東京都品川区大崎2丁目1番17号

(72) 発明者 吉田 幹生

東京都品川区大崎2丁目1番17号 株式会
社明電舎内

(72) 発明者 菅谷 勝洋

東京都品川区大崎2丁目1番17号 株式会
社明電舎内

(74) 代理人 弁理士 志賀 富士弥 (外1名)

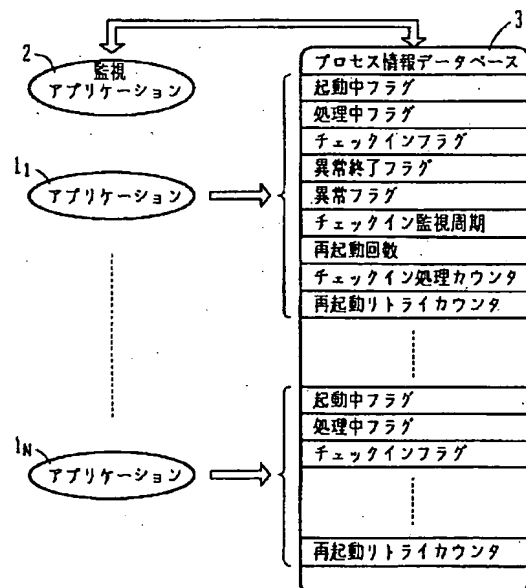
(54) 【発明の名称】 ソフトウェアの異常監視方式

(57) 【要約】

【課題】 コンピュータに搭載するアプリケーションは、プロセスループやプロセスハングには自身で異常検出ができない。

【解決手段】 データベース3は、監視対象アプリケーション毎に、その起動中・処理中・監視中をそれぞれ表す起動中フラグと処理中フラグとチェックインフラグと、監視対象アプリケーションの異常を判断するチェックイン監視周期と、監視処理回数を積算するチェックイン処理カウンタ等を監視情報とし、監視対象アプリケーション1₁~1_Nは、起動時、処理中に応じて各フラグをセット・リセットし、監視アプリケーション2は、チェックインフラグ等の参照とチェックインカウンタのカウントアップやクリア等で監視対象アプリケーションの異常を検出する。また、異常検出後のアプリケーションの再起動メカニズムにより一過性の異常に対する自動復帰を可能とする。

異常監視システムの構成図



【 特許請求の範囲】

【 請求項1 】 コンピュータシステムに搭載する各アプリケーションの異常を監視するにおいて、

監視対象アプリケーション毎に、その起動中・処理中・監視中をそれぞれ表す起動中フラグと処理中フラグとチェックインフラグと、監視対象アプリケーションの異常を判断するチェックイン監視周期と、監視処理回数を積算するチェックイン処理カウンタとを監視情報として持つ監視情報記憶手段と、

前記起動中フラグを起動時にセットし、前記処理中フラグを処理開始時にセットして処理終了時にリセットし、前記チェックインフラグを処理開始時にセットする監視対象アプリケーションと、

前記監視対象アプリケーション毎に、前記監視情報記憶手段のチェックイン監視周期で前記チェックインフラグを参照し、該チェックインフラグがセットされているときに当該監視対象アプリケーションが処理中として該チェックインフラグをリセットしかつ前記チェックイン処理カウンタをクリアし、前記処理中フラグがセットされかつ前記チェックインフラグがリセットされた状態で前記チェックイン処理カウンタをカウントアップし、該カウンタの値が前記チェックイン監視周期を越えたときに当該監視対象アプリケーションを異常と判定する監視アプリケーションとを備えたことを特徴とするソフトウェアの異常監視方式。

【 請求項2 】 前記監視情報記憶手段は、監視対象アプリケーション毎に、その異常終了を表す異常終了フラグと、監視対象アプリケーションの異常検出時の再起動回数が設定される再起動回数と、再起動回数を積算する再起動リトライカウンタと、前記再起動回数を越えて異常が発生したことを表す異常フラグとを監視情報として設け、

前記監視対象アプリケーションは、そのタスクが異常終了したときに前記異常終了フラグをセットし、前記監視アプリケーションは、監視対象アプリケーションを異常と判定したときに当該監視対象アプリケーションを強制停止し、当該監視対象アプリケーションの前記再起動リトライカウンタが前記再起動回数に満たないときには該再起動リトライカウンタをカウントアップして当該監視対象アプリケーションを再起動し、該再起動リトライカウンタが再起動回数を越えたときに前記異常フラグをセットして当該監視対象アプリケーションの縮退運転に遷移することを特徴とする請求項1に記載のソフトウェアの異常監視方式。

【 発明の詳細な説明】

【 0001 】

【 発明の属する技術分野】 本発明は、監視制御システム等を構成するコンピュータシステムの異常監視方式に係り、特にコンピュータに搭載した各アプリケーションの異常を外部から監視するためのソフトウェアの異常監視

方式に関する。

【 0002 】

【 従来の技術】 コンピュータシステムは、多くのアプリケーションソフトウェアが搭載されて監視制御システムなどを構築する。監視制御システムなど、高い信頼性を要求されるコンピュータシステムではそのソフトウェアは如何なる場合においてもシステムダウンを引き起こしてはならないが、現実にはソフトウェアの問題によるシステムダウンが発生することがある。このため、システムの各機能を司るアプリケーションの異常検出方式が非常に重要となってくる。

【 0003 】 現在、監視制御システムに搭載するアプリケーションで発生するソフトウェア異常の種類及びその時の異常検出方法は、以下のようなものがある。

【 0004 】 (1) ソフトウェア異常の種類

OS の中心に位置するカーネルが要因のレベルと、アプリケーションのレベルに原因がある場合があるが、以下に示すような分類とする。

【 0005 】 (a) システムコールエラー…システムサービス (C 標準関数等) 発行時のエラー。

【 0006 】 (b) I / O エラー…デバイス、ファイル等のアクセス時のエラー。

【 0007 】 (c) 例外…アクセスバイオレーション・整数オーバフローなど (配列外参照も含む) 。

【 0008 】 (d) プロセスハング…I / O の要求完了待ちなどの要因でプロセスが処理を継続できない状態。

【 0009 】 (e) プロセसरूप…アルゴリズムやデータ不良によりプロセスが起動要因なしに永久にループする状態。

【 0010 】 (f) データ入力エラー…処理を行なうために必要なファイルやデータベースの異常。

【 0011 】 (2) 異常検出方法

(a) システムコールエラー…システムコールのエラーは、リターンコードで判別する。C 言語レベルのエラーもシステムコールエラーと同様である。

【 0012 】 (b) I / O エラー…I / O エラーはシステムコールエラーと同様に、システムコールのリターンステータスでエラーを検出する。

【 0013 】 (c) 例外…OS が検出してプロセス自身 (バンドラ) に通知される。

【 0014 】 (d) データエラー・起動データエラー…起動データエラーと処理における入力・加工データエラーに分けられる。どちらもユーザプロセスのチェックアルゴリズムで検出する。また、重要なデータについては、チェックサムなどのセキュリティデータを付加して、定期的にあるいはアクセス時にチェックする方法がある。

【 0015 】 以上のように、現在は、アプリケーション自身にて判断できる異常検出方法はそれぞれ確立されているが、プロセスハング及びプロセसरूपなどアプリ

3

ケーション自身にて判断できない異常検出方法は確立されていない。

【0016】

【発明が解決しようとする課題】アプリケーション自身にて判断できる異常検出においては、異常処理（異常情報の保存等）を実行し異常復帰を行い、システムダウンに至らないよう回避することができる。

【0017】しかし、アプリケーションで判断できないプロセスループ及びプロセスハングについては異常復帰ができない。

【0018】また、プロセスループではループする場所（I/Oを含んだ大きなループになる場合やCPUバウンドで数ステップを繰り返し実行し続ける場合）により、資源を占有している場合や優先順位が高い場合は他のプロセスの処理に大きな影響を及ぼし、最終的にはシステムダウンを招く恐れもある。

【0019】本発明の目的は、アプリケーションを外部から監視しながらアプリケーションのプロセスループやプロセスハングの異常検出ができる異常監視方式を提供することにある。

【0020】

【課題を解決するための手段】本発明は、アプリケーションのプロセスループやハング等の異常を外部から疎結合で監視し、さらに異常検出後のアプリケーションの再起動メカニズムにより一過性の異常に対する自動復帰を可能とするため、監視メカニズムをデータベース等に設けた各種フラグとカウンタを利用して実現するもので、以下の方式を特徴とする。

【0021】コンピュータシステムに搭載する各アプリケーションの異常を監視するにおいて、監視対象アプリケーション毎に、その起動中・処理中・監視中をそれぞれ表す起動中フラグと処理中フラグとチェックインフラグと、監視対象アプリケーションの異常を判断するチェックイン監視周期と、監視処理回数を積算するチェックイン処理カウンタとを監視情報として持つ監視情報記憶手段と、前記起動中フラグを起動時にセットし、前記処理中フラグを処理開始時にセットして処理終了時にリセットし、前記チェックインフラグを処理開始時にセットする監視対象アプリケーションと、前記監視対象アプリケーション毎に、前記監視情報記憶手段のチェックイン監視周期で前記チェックインフラグを参照し、該チェックインフラグがセットされているときに当該監視対象アプリケーションが処理中として該チェックインフラグをリセットしかつ前記チェックイン処理カウンタをクリアし、前記処理中フラグがセットされかつ前記チェックインフラグがリセットされた状態で前記チェックイン処理カウンタをカウントアップし、該カウンタの値が前記チェックイン監視周期を越えたときに当該監視対象アプリケーションを異常と判定する監視アプリケーションとを備えたことを特徴とする。

4

【0022】また、前記監視情報記憶手段は、監視対象アプリケーション毎に、その異常終了を表す異常終了フラグと、監視対象アプリケーションの異常検出時の再起動回数が設定される再起動回数と、再起動回数を積算する再起動リトライカウンタと、前記再起動回数を越えて異常が発生したことを表す異常フラグとを監視情報として設け、前記監視対象アプリケーションは、そのタスクが異常終了したときに前記異常終了フラグをセットし、前記監視アプリケーションは、監視対象アプリケーションを異常と判定したときに当該監視対象アプリケーションを強制停止し、当該監視対象アプリケーションの前記再起動リトライカウンタが前記再起動回数に満たないときには該再起動リトライカウンタをカウントアップして当該監視対象アプリケーションを再起動し、該再起動リトライカウンタが再起動回数を越えたときに前記異常フラグをセットして当該監視対象アプリケーションの縮退運転に遷移することを特徴とする。

【0023】

【発明の実施の形態】図1は、本発明の実施形態を示すソフトウェア異常監視システムの構成図であり、監視制御システムを構成する多数のアプリケーション1₁～1_Nを監視アプリケーション2が外部から監視することでソフトウェアの異常を検出する。

【0024】このソフトウェア異常検出（以下、チェックイン機能と呼ぶ）は、アプリケーションが正常に動作していることを周期的に外部より監視し、プロセスループ、プロセスハング等の異常を検出した時、当該アプリケーションの強制停止及び再起動又は縮退運転への遷移を行う。

【0025】監視アプリケーション2が各監視対象アプリケーション1₁～1_Nを周期的に外部より監視（以下、チェックイン処理と呼ぶ）するため、監視情報記憶手段としてのプロセス情報データベース3には、以下のデータ（フラグやカウンタ）を各監視対象アプリケーション毎に用意する。

【0026】・起動中フラグ…監視対象アプリケーションが起動中であるかを表す。

【0027】・処理中フラグ…監視対象アプリケーションが処理中であるかを表す。

【0028】・チェックインフラグ…監視対象アプリケーションの監視中であるかを表す。

【0029】・異常終了フラグ…監視対象アプリケーションが異常終了したかを表す。

【0030】・異常フラグ…監視対象アプリケーションが再起動回数を満了して異常終了したかを表す。

【0031】・チェックイン監視周期…監視対象アプリケーションの異常を判断する監視周期を表す。この周期はアプリケーションの各タスク毎に設定される。

【0032】・再起動回数…監視対象アプリケーションを再起動する回数を表す。

50

5

【0033】・チェックイン処理カウンタ…監視対象アプリケーションの監視を実施した回数を積算するカウンタ。

【0034】・再起動リトライカウンタ…監視対象アプリケーションを再起動した回数を積算するカウンタ。

【0035】以上の情報のうち、各監視対象アプリケーションに対応付けた各フラグはデータ上でビット扱いとし、チェックイン周期やカウンタは数値データとして扱われる。

【0036】図1における各監視対象アプリケーション11～18は、データベース3の各フラグを以下のタイミングでセットする。

【0037】・起動中フラグ…当該アプリケーションが起動時にセット。

【0038】・処理中フラグ…当該アプリケーションが *

6

*処理開始時にセットし、処理終了時にリセット。

【0039】・チェックインフラグ…当該アプリケーションが処理開始時にセット。

【0040】・異常終了フラグ…当該アプリケーションが異常終了時にセット。

【0041】一方、監視アプリケーション2は、監視周期にてデータベース3の各フラグを参照し、図2に示す異常監視処理フローに従って異常監視と異常検出を行い、フラグのセット又はリセットを行う。このときの各フラグのセット、リセットは、下記表1に示す。この表中、各APLは、各監視対象アプリケーションを意味し、監視APLは監視アプリケーションを意味する。

【0042】

【表1】

概要	フラグ(*2)	セット	リセット	参照
			通常	
各タスクが起動した	起動中フラグ	各APL	監視APL	監視APL
各タスクが処理中	処理中フラグ	各APL	各APL 監視APL	監視APL
各タスクが処理を終了した	処理中フラグ	各APL	各APL	
各タスクが異常終了した	異常終了フラグ	各APL	監視APL	監視APL
タスク異常	異常フラグ	監視APL	監視APL	監視APL

【0043】図2において、監視アプリケーションによる異常監視処理は、チェックイン監視周期で処理中フラグとチェックインフラグを参照し、それらのいずれかがセットされているか否かを判定する(S1)。

【0044】この判定で、いずれかがセットされており、それがチェックインフラグであるとき(S2)、当該アプリケーションは正常に処理中であるとみなしてチェックインフラグのみをリセットし(S3)、チェックインカウンタをクリアする(S4)。

【0045】判定処理S2の判定において、チェックインフラグがセットされていないとき、すなわち処理中フラグのみがセットされているとき、この状態をチェックイン監視中として現在状態がチェックイン処理カウンタの値がチェックイン監視周期以上か未満かをタスク毎にチェックし(S5)、チェックイン監視周期未満ではチェックイン処理カウンタをカウントアップしておく(S6)。

【0046】判定処理S5の判定において、処理中フラグのみがセット状態でチェックイン監視周期以上になったとき、監視アプリケーション2は当該監視対象アプリ

ケーションが異常であると判断し、監視対象アプリケーションの当該タスクの実行を強制停止させる(S7)。

【0047】以上までのチェックイン処理により、監視対象アプリケーションで判断できないプロセスループやプロセスハングについてその異常検出ができる。

【0048】次に、上記のタスクの強制停止に対して、当該監視対象アプリケーションは、データベース3の異常終了フラグをセットする。この異常終了フラグのセットに対して、監視アプリケーション2は、データベース3の再起動回数と再起動リトライカウンタを参照し、当該タスクは再クリエイト対象か否かを判定する(S8)。

【0049】この判定処理S8において、再起動リトライカウンタの値が再起動回数に満たない場合、再起動リトライカウンタをカウントアップすると共にチェックイン処理カウンタをリセットし(S9)、当該アプリケーションのタスクの再クリエイトを行い(S10)、当該アプリケーションの再起動を行う。これにより、異常検出後のアプリケーションの再起動ができ、一過性の異常に対する自動復帰が可能となる。

7

【0050】また、判定処理S8において、再起動リトライカウンタの値が再起動回数を満了した場合、監視アプリケーション2は異常フラグをセットし(S11)、当該アプリケーションを異常扱いのままとして縮退運転に遷移する。すなわち、当該アプリケーション及び関連アプリケーションの強制停止を行う。

【0051】以上のように、監視アプリケーションによる監視対象アプリケーションに対する外部からの異常監視処理により、アプリケーションで判断できないプロセスループ及びプロセスハングについて異常検出が可能となり、異常の対処(異常情報の保存及び異常アプリケーションの強制終了、再起動または縮退運転等)を行うことができる。

【0052】また、プロセスループによるシステムへの影響もシステムダウンに至る前に未然に防ぐことが可能となり、ソフトウェアが原因となるシステムダウンを無くすことができる。

【0053】このような異常監視処理における監視対象アプリケーション側の処理は、図3～図5に示すように、監視対象アプリケーションの実行形態に応じてキュー単位やプロセス単位の異常監視を各フラグのセット、リセット機能で実現し、システム資源の影響による監視不能を排除する。

【0054】図3は、監視対象アプリケーションが永久起動プロセスの場合の処理を示す。この監視対象アプリケーションにおいては、システム共通域のプロセス初期化処理として、コンディションハンドラ登録やEXITハンドラ登録、キューターミナルの初期化などを行い(S21)、さらにプロセス固有部の初期化を行い(S22)、データベース3の起動中フラグをセットし(S23)、この後に永久起動に入る。なお、起動中フラグのセットでエラーが発生したときはデータベース3の異常終了フラグのセットにより監視アプリケーション2へ通知し、自らのプロセス起動を停止する。

【0055】上記のプロセスの永久起動では、データベース3の処理中フラグをクリアし(S24)、キュー情報を取得し(S25)、データベース3のチェックインフラグをセットし(S26)、処理中フラグをセットし(S27)、プロセスロック中でないとき(S28)に各個別キューの処理を行う(S29)。

【0056】図4は、監視対象アプリケーションが起動終了プロセスの場合の処理を示す。処理S21～S23の部分は図3の場合と同様となる。起動中フラグのセット(S23)の後、データベース3のチェックインフラ

8

グをセットし(S31)、処理中フラグをセットし(S32)、プロセスロック中でないとき(S33)に各個別プロセスの処理を行う(S34)。

【0057】プロセス処理を終了した後、データベースの処理中フラグをクリアし(S35)、起動中フラグをクリアしてプロセスを終了する(S36)。

【0058】図5は、監視対象アプリケーションが特殊イベント(ウインドウイベント、RPCイベントなど)で起動するプロセスの場合の処理を示す。同図が図3の永久起動プロセスの処理と異なる部分は、処理中フラグのクリア(S24)後にイベント発生を待ち(S30)、イベント発生でチェックインフラグ及び処理中フラグのセット(S26、S27)を行い、発生したイベントに対するプロセスを個別処理する(S34)。

【0059】この特殊イベント処理では、永久起動プロセスの場合と同様に、チェックインのイベントの代わりに模擬イベントを発生させる方法もあるが、ここではチェックインフラグによる監視を行う。

【0060】

【発明の効果】以上のとおり、本発明によれば、データベース等に設けた各種フラグとカウンタを利用して監視対象アプリケーションを外部より疎結合で監視し、その異常検出を行うようにしたため、監視対象アプリケーションで判断できないプロセスループやプロセスハングについて異常検出が可能となり、異常情報の保存及び異常アプリケーションの強制終了、再起動または縮退運転等を行うことができる。

【0061】また、プロセスループによるシステムへの影響もシステムダウンに至る前に未然に防ぐことが可能となり、ソフトウェアが原因となるシステムダウンを無くすことができる。

【図面の簡単な説明】

【図1】本発明の実施形態を示す異常監視システムの構成図。

【図2】実施形態における異常監視処理フロー。

【図3】実施形態における永久起動プロセスの処理フロー。

【図4】実施形態における起動終了プロセスの処理フロー。

【図5】実施形態における特殊イベント処理フロー。

【符号の説明】

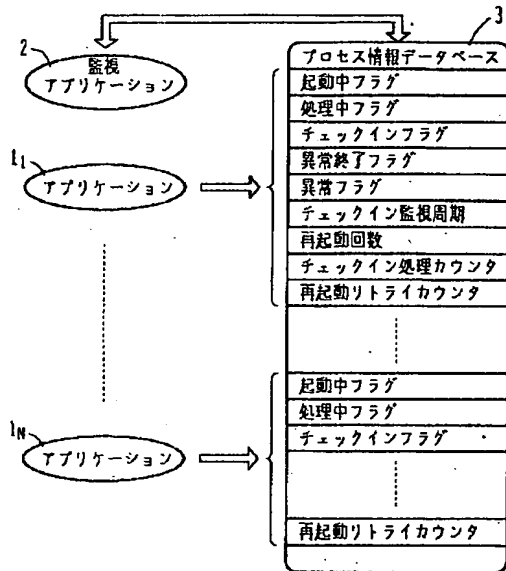
1、1N…監視対象アプリケーション

2…プロセス情報データベース

3…監視アプリケーション

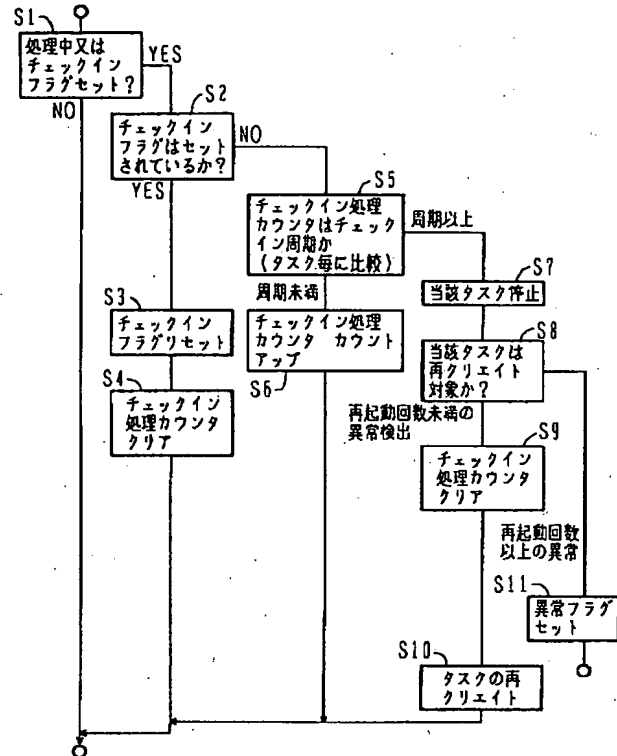
【 図1 】

異常監視システムの構成図



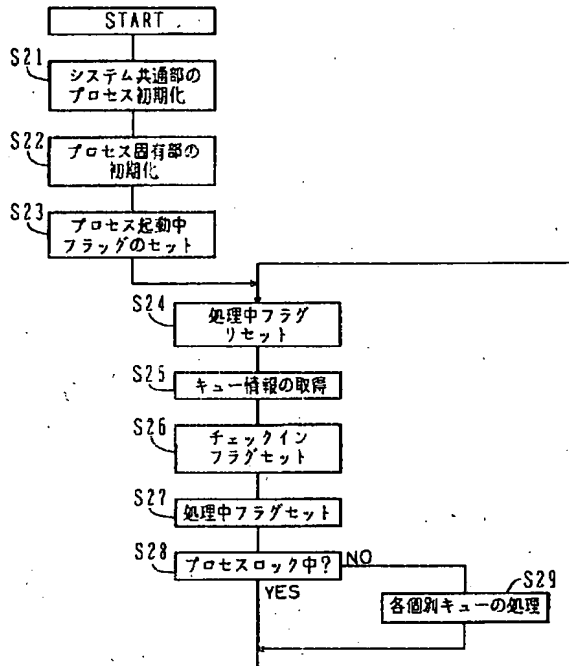
【 図2 】

異常監視処理フロー



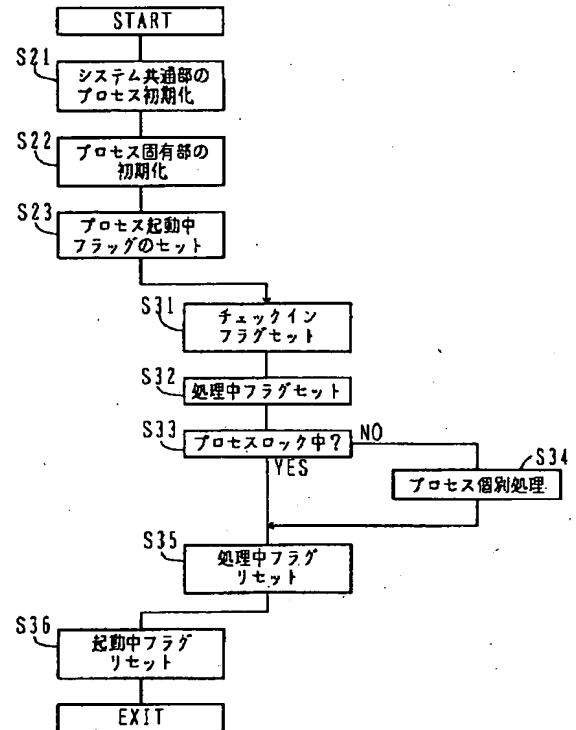
【 図3 】

永久起動プロセスの処理フロー



【 図4 】

起動終了プロセスの処理フロー



【 図5 】

特殊イベント処理フロー

